anticipated by Forsman et al., U.S. Pat. No.: 6,665,813 (hereinafter, "Forsman"). This rejection is respectfully traversed.

Independent Claim 1

A proper §102 rejection requires that a single reference anticipate the claimed invention as a whole. Each and every element recited in the claimed invention must be disclosed by this single reference.

The Examiner asserts that Claim 1 discloses:

> "defining in the storage device information related to a first data structure [composite code 308, Fig 3] within a plurality of copies of a second data structure [recovery code 304 - copy A, Fig 3, recovery code 306 - copy B, Fig 3]
>
> rebuilding information related to the first data structure using a copy of one of the plurality of copies of the second data structure upon corruption of another copy of the plurality of copies of the second data structure [col 6, lines 5-15]".

Applicants respectfully disagree with the Examiner's application of Forsman to Claim 1 for at least two reasons. First, Forsman at col. 6, lines 4-15, and Fig. 3 does not disclose each and every element of the recited claim as required under a proper §102 rejection. Second, the boot mechanism described in Forsman (col. 5 - col. 6, lines 4-15, and Fig. 3) not only fails to anticipate claim 1 but also teaches away from the recited elements.

Forsman at col. 5, lines 44-55, discloses a process for correcting recovery code Copy A or Copy B upon corruption of either Copy A or Copy B. If Copy A is "detected to be

corrupted...then the duplicate copy of the recovery code in Copy B...is copied into Copy A". Forsman, col. 5, lines 48-51. Similarly, Copy B is checked for corruption and if found to be corrupted, "then the recovery code in Copy A is copied into Copy B". Id. at col. 5, lines 53-55.

Claim 1 differs from Forsman because claim 1 recites that "the information related to the first data structure" is rebuilt using a copy of one of the plurality of copies of the second data structure upon corruption of another copy of one of the plurality of copies of the second data structure. Forsman does not disclose these recited elements because Forsman only discloses rebuilding Copy A when Copy A is corrupted rather than when Copy B is corrupted. In the alternative, Forsman rebuilds Copy B when Copy B is corrupted rather than when Copy A is corrupted. Thus, Forsman discloses restoring a piece of data when the same piece of data is corrupted, which differs from the elements recited in claim 1.

Moreover, Forsman at col. 5, lines 57-67, distinguishes itself from claim 1 by disclosing that upon corruption of the composite code, the composite code is restored using the recovery code to obtain a "fresh copy of the composite code". Thus, Forsman discloses that code, the "composite code" in Forsman, is restored upon corruption of the same piece of code. Forsman, col. 5, lines 57-64.

Claim 1, however, differs from Forsman because claim 1 recites that "the information related to the first data structure" is rebuilt using a copy of one of the plurality of copies of the second data structure upon corruption of another copy of one of the plurality of copies of the second data structure. Forsman does not disclose these elements because it rebuilds the "composite code" when the composite code is corrupted. Essentially, Forsman discloses restoring a piece of code (e.g., the composite code) when the same piece of code is corrupted,

which differs from the elements recited in claim 1.

Since a proper §102 rejection requires that each and every element disclosed in a claim must be shown by a single piece of cited prior art, Forsman fails to anticipate claim 1, rendering claim 1 allowable.

Dependent Claim 2

The argument set forth above is equally applicable to dependent claim 2. Since the base claim is allowable, the dependent claim must also be allowable.

In addition, Forsman discloses at col. 4, lines 33-53, a Copy A and a Copy B. If Copy A is corrupted during an update, then the contents of Copy B are written to the location of Copy A. In the alternative, if Copy B is corrupted during an update, then the contents of Copy A are written to the location of Copy B. Claim 2, however, recites "defining the information related to a *first* data structure" and "updating a plurality of copies of the *second* data structure prior to the write operation." (emphasis added). Consequently, since Forsman fails to recite each and every element of Claim 2, it is allowable for this additional reason as well.

Dependent Claim 3

The argument set forth above is equally applicable to dependent claim 3. Since the base claim is allowable, the dependent claim must also be allowable.

In addition, the Applicants respectfully disagree that Forsman discloses, at col. 6, lines 28-38, "differentiating which of the plurality of copies of the second data structure has the most recent data." Forsman simply discloses that it uses an uncorrupted version of the recovery code

if an update to one of the copies (Copy A or Copy B) is corrupted during an update. Thus, Forsman does not teach or suggest differentiating between the most recent data but differentiating between corrupted and uncorrupted data. Since a proper §102 rejection requires that each and every element of a rejected claim must be disclosed by a single reference, Applicants therefore believe claim 3 is patentable over Forsman for this additional reason.

Dependent Claims 4-5

The argument set forth above is equally applicable to dependent claims 4 and 5. Since the base claim is allowable, the dependent claim must also be allowable.

Dependent Claim 6

Claim 6 ultimately depends on independent claim 1 and thus, the argument set forth above is equally applicable to dependent claim 6. Since the base claim is allowable, the dependent claim must also be allowable.

In addition, Applicants respectfully disagree because Forsman does not disclose or suggest a method wherein information related to one data structure is rebuilt "after data corruption of one of the plurality of copies of the *second* data structure," (emphasis added), as recited in claim 6.

Specifically, Forsman at col. 5, lines 44-55, discloses a process for correcting Copy A or Copy B of the recovery code upon corruption of either Copy A or Copy B. If Copy A is "detected to be corrupted...then the duplicate copy of the recovery code in Copy B...is copied into Copy A". Forsman, col. 5, lines 48-51. Similarly, Copy B is checked for corruption and if found to be corrupted, "then the recovery code in Copy A is copied into Copy B". Id. at col. 5,

lines 53-55. Forsman also discloses that the first piece of code, the "composite code" in Forsman, is restored using a second piece of code, which is disclosed in Forsman as the "recovery code", upon corruption of the first piece of code.

Forsman consequently fails to disclose that "rebuilding the information related to the *first* data structure includes using one or more of the plurality of copies of the second data structure to find other file management structures after data corruption of one of the plurality of copies of the *second* data structure", (emphasis added), as recited in claim 6. Forsman does not disclose these elements because it rebuilds the first piece of code (the "composite code") when the first piece of code is corrupted. Since a proper §102 rejection requires that each and every element disclosed in a claim must be shown by a single piece of cited prior art, Forsman fails to anticipate Claim 6 for this additional reason, rendering Claim 6 allowable.

Dependent Claim 22

The argument set forth above is equally applicable here. Since the base claim is allowable, the dependent claim must also be allowable.

Moreover, Forsman at col. 4, lines 18-32, describes the use of boot code in a computer system claim 22 and the different approaches to storing the boot code. It neither teaches nor discloses the elements recited in Claim 22, including the element of "associating a memory address to the first data structure". Consequently, Forsman fails to anticipate claim 22, rendering claim 22 allowable for this additional reason.

Dependent Claim 23

Claim 23 ultimately depends on claim 1 and thus, the arguments above are equally

applicable here. Since the base claim is allowable, the dependent claim must also be allowable.

Independent Claim 8

For reasons similar to those explained above for claim 1, Forsman does not disclose the above elements of claim 8 because Forsman restores the "composite code" when the composite code is corrupted. Thus, Forsman restores code when the same code is corrupted.

Claim 8, however, recites "rebuilding the memory location of the data structure using the plurality of base block copies in the event of data corruption of one of the base block copies." Thus, Forsman fails to disclose the present invention as claimed. Since a proper §102 rejection requires that each and every element of a rejected claim must be disclosed by a single reference, claim 8 must be found allowable over Forsman.

Dependent Claims 9-12

Claims 9-12 are ultimately dependent on claim 8 and thus, the arguments above are equally applicable here. The base claim being allowable, the dependent claims must also be allowable.

Independent Claim 15

For reasons similar to those explained above for claims 1 and 8, Forsman does not disclose claim 15 because Forsman restores a piece of code (e.g., the "composite code") when the same piece of code is corrupted. Claim 15, however, recites "rebuilding the location of the data structure in the event one base block copy cannot be located or verified by using another base block copy." Thus, Forsman fails to disclose the present invention as claimed because

Forsman disclosing restoring the composite code when the composite code, rather than the recovery code, is corrupted. Since a proper §102 rejection requires that each and every element of a rejected claim must be disclosed by a single reference, claim 15 must be found allowable over Forsman.

Dependent Claims 16-21

Claims 16-21 are ultimately dependent on allowable claim 15 and thus, the arguments above are equally applicable here. The base claim being allowable, the dependent claims must also be allowable.

The Second 35 U.S.C. §102 Rejection

Claims 15-19 stand rejected under 35 U.S.C. §102(b) as being allegedly anticipated by Froemke et al., U.S. Pat. No.: 5,239,640 (hereinafter, "Froemke"). This rejection is respectfully traversed.

Independent Claim 15

The Examiner rejects claim 15 on the assertion that Froemke discloses "defining in the non-volatile memory a location of a data structure with at least two base block copies [col 1, lines Fig 1, 22 and 26, Fig 2, 30-36], and rebuilding the location of the data structure in the event one base block copy cannot be located or verified by using another base block copy [col 4, lines 28-36]".

Applicants respectfully disagree. Froemke does not disclose the use of a base block and or the use of copies. Elements 22 and 26 in Fig. 1 are not copies or each other because they hold

different types of data. Write staging area, element 26, holds both write commands and data, while element 22 only holds data. Froemke col. 4, lines 3-9. Since these elements do not hold the same kind of data, they cannot be considered copies of each other, rendering the §102 rejection against claim 15 improper.

Further, element 22 is a DASD and comprised of elements 30-36 in Fig. 3. Froemke discloses that if one of the elements (30, 32, 34 and 36) in the DASD fails, an XOR operation can be performed between the checksum stored in checksum storage 24 and the remaining source data that can still be read from the remaining elements of the DASD assembly that have not failed. Id., col. 5, lines 23-29 and Table 2. The data in the failed element comprising part of the DASD assembly is reconstructed by performing an XOR operation between the checksum and the remaining source data. Thus, Froemke discloses a storage technique that uses checksum data and XOR operations to rebuild source data without the use of copies. The invention as recited in claim 15 does not use this technique, and hence, Froemke fails to anticipate claim 15 for this additional reason, rendering claim 15 allowable.

In addition to the above, Applicants wish to make re-emphasize their argument made in the prior response to office action in the event of an appeal. Elements 22 and 26 are not copies of each other because of the preempt function. Froemke at col. 4, lines 37-52, and col. 6, lines 14-17, and col. 8, lines 32-44. For this additional reason, Froemke fails to anticipate claim 15, rendering claim 15 allowable.

Dependent Claims 16, 17 and 19

Claims 16, 17 and 19 ultimately depend on claim 15 and thus, the arguments above are equally applicable here. The base claim being allowable, the dependent claims must also be

allowable.

Dependent Claim 18

Claim 18 ultimately depends on claim 15 and thus, the arguments stated for claim 15 are equally applicable here. Since the base claim is allowable, the dependent claim must also be allowable.

In addition, the Examiner asserts that Froemke at col. 4, lines 37-43, discloses "defining the location of the data structure includes selecting each of the base copies so at most one can be corrupted." Applicants respectfully disagree. Froemke only discloses the use of a preempt function. It does not disclose avoiding corruption but preempting writes that would otherwise occur in the DASD assembly, thereby reducing write latency. Id. At col. 4, lines 47-51. See also, Id. at col. 8, lines 40-44. This rationale for reduced write latency not only fails to disclose the elements of claimed invention but also teaches away from it. For this additional reason, claim 18 should be found allowable.

The First 35 U.S.C. §103 Rejection

Dependent claims 7, 13, 14, 20, and 21 stand rejected under 35 U.S.C. §103(a) as being allegedly unpatentable over Forsman et al., in view of See (U.S. Pat. No.: 6,170,066). This rejection is respectfully traversed.

Generally

Applicants respectfully observe that it is not clear if the Examiner is also citing Froemke as a basis for this §103 rejection. The Examiner uses Froemke instead of Forsman as a basis for

this rejection but combines Forsman and See when actually asserting the § 103 rejection in the Office Action. Applicants have made the response below to this rejection for claims 1, 7, 8, 13, 14, 15, 20 and 21 solely to provide a responsive reply to the Office Action. Applicants, however, believe they cannot fully respond to this rejection since it is not clear which of the references are applicable to the rejected claims.

Independent Claims 1, 8 and 15

Froemke fails to teach or suggest the elements recited in independent claims 1, 8 and 15, respectively, for the reasons stated in the Applicants' prior response to office action. In the alternative, Forsman also fails to teach or suggest the elements recited in independent claims 1, 8 and 15 for the reasons stated above, respectively, rendering independent claims 1, 8 and 15 allowable.

Dependent Claims 7, 14 and 21

Claims 7, 14 and 21 depend on allowable claims 1, 8 and 15, respectively, and thus, should be allowable because they ultimately depend on allowable independent claims.

Further, See does not teach using pre-erased recovery blocks. See instead discloses delayed erase operations. See delays erasing an entire block in order to avoid "erasing all of the valid information that remains in the block along with invalid information." See, col. 1, lines 62-65. See discloses a process of "writing new information to a new memory area rather than" writing "over the old data; and the old data is marked invalid. Then, *after a sufficient portion of a block has been marked invalid*, all valid information remaining in the block is written to the new memory area; and the entire block may then be erased, typically using a background process."

See, col. 1, line 66 - col. 2, line 3. (emphasis added). Consequently, See fails to teach or suggest the use of "pre-erased" recovery blocks as recited in the present invention.

Since a proper §103 rejection requires that the recited references teach or suggest the combination of elements in a rejected claim, Applicants therefore believe that claims 7, 14 and 21 are patentable over the cited references.

Dependent Claims 13 and 20

Claims 13 and 20 are ultimately dependent on claims 8 and 15, respectively, and thus, the arguments above are equally applicable here. The base claims being allowable, the dependent claims must also be allowable.

Further, the Examiner states that "Forsman discloses the essential elements of the claimed invention except for pointers to other data structures. See '066 discloses pointers to other data structures [col 7, lines 12-15]. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Forsman to include pointers to other data structures as taught by See '066 for the purpose of indicating an empty data structure." Applicants respectfully disagree. A proper §103 rejection requires that the recited references teach or suggest the combination of elements in a rejected claim. Neither Forsman nor See teach or suggest the combination of the method for recovering data in a memory as recited in claim 15 and pointers to other data structures. Thus, it would not have been obvious to one of ordinary skill in the art to combine Forsman with See. Even if Forsman and See were combined, See only suggests the use of pointers for the purpose of indicating full data blocks. Nothing in See teaches or suggests the embodiments of the present invention because the present invention is not limited to the use of pointers for the purpose of indicating full data blocks.

Since a proper §103 rejection requires that the recited references teach or suggest the combination of elements in a rejected claim, Applicants therefore believe that claims 13 and 20 are patentable over the cited references.

The Second 35 U.S.C. §103 Rejection

Dependent claims 20, and 21 stand rejected under 35 U.S.C. §103(a) as being allegedly unpatentable over Froemke et al., in view of See (U.S. Pat. No.: 6,170,066). This rejection is respectfully traversed.

Independent Claim 15

The Examiner states that "Froemke discloses the essential elements of the claimed invention except for pointers to other data structures. See '066 discloses pointers to other data structures [col 7, lines 12-15]. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify Froemke to include pointers to other data structures as taught by See '066 for the purpose of indicating an empty data structure [col 7, line 12-15]." Applicants respectfully disagree.

Applicants respectfully disagree because there is no teaching or suggestion in either Froemke or See to support their combination. See does not use or relate to the inventive concepts, such as using a preempt function combination with a write staging storage area and DASD assembly, disclosed in Froemke. Consequently, the combination of these references as a basis for the §103 rejection is improper and is made in hindsight, rendering claim 20 allowable.

Moreover, even if combined, Froemke in view of See fail to disclose all of the elements recited in claim 15 because of the following reasons. First, Froemke does not disclose the use of

a base block and or the use of copies. Elements 22 and 26 in Fig. 1 are not copies or each other because they hold different types of data. Write staging area, element 26, holds both write commands and data, while element 22 only holds data. Froemke col. 4, lines 3-9. Since they do not hold the same kind of data, these elements cannot be considered copies.

Second, element 22 is a DASD and is comprised of elements 30-36 in Fig. 3. Froemke discloses that if one of the elements (30, 32, 34 and 36) in the DASD fails, an XOR operation can be performed between the checksum stored in checksum storage 24 and the remaining source data that can still be read from the remaining elements of the DASD assembly that have not failed. Id., col. 5, lines 23-29 and Table 2. The data in the failed element comprising part of the DASD assembly is reconstructed by performing an XOR operation between the checksum and the remaining source data. Thus, Froemke discloses a storage technique that uses checksum data and XOR operations to rebuild source data without the use of copies.

And last, elements 22 and 26 are not copies of each other because of the preempt function disclosed in Froemke at col. 4, lines 37-52 and col. 6, lines 14-17 and col. 8, lines 32-44.

Consequently, for the reasons stated above, claim 15 should be found allowable.

Dependent Claim 20

Claim 20 ultimately depends on claim 15 and thus, the arguments stated for claim 15 are equally applicable here. Since the base claim is allowable, the dependent claim must also be allowable.

Moreover, claim 20 recites, among other things, "wherein the data structure includes pointers to other data structures selected from a group consisting of remap information, wear-

leveling tables, configuration data, recovery information, and a combination thereof". See only teaches or suggests the use of pointers for the purpose of indicating full data blocks. Nothing in See teaches or suggests the claimed invention because the present invention is not limited to the use of pointers for the purposes of indicating full data blocks. Hence, for these additional reasons, claim 20 should be found allowable.

Dependent Claim 21

Claim 20 ultimately depends on claim 15 and thus, the arguments stated for claim 15 are equally applicable here. Since the base claim is allowable, the dependent claim must also be allowable.

Moreover, claim 21 recites, among other things, "wherein the non-volatile memory includes pre-erased recovery blocks; and the base block copies are written to pre-erased recovery blocks". See, however, does not teach using pre-erased recovery blocks. See instead discloses delayed erase operations. See delays erasing an entire block in order to avoid "erasing all of the valid information that remains in the block along with invalid information." See, col. 1, lines 62-65.

See discloses a process of "writing new information to a new memory area rather than" writing "over the old data; and the old data is marked invalid. Then, *after a sufficient portion of a block has been marked invalid*, all valid information remaining in the block is written to the new memory area; and the entire block may then be erased, typically using a background process." See, col. 1, line 66 - col. 2, line 3. (emphasis added). Consequently, See fails to teach or suggest the use of "pre-erased" recovery blocks as recited in present invention, rendering claim 21 allowable for this additional reason.

It is believed that the above-identified patent application is now in condition for allowance.

If, in the opinion of the Examiner, an interview would expedite the prosecution of this application, the Examiner is invited to call the undersigned attorney at the number indicated below.

Respectfully submitted,
BiTMICRO Networks, Inc.

Dated: October 14, 2004

Shabneez Kotadia
Reg. No. 53,153

BiTMICRO Networks, Inc.
45550 Northport Loop East
Fremont, CA 94538
(510) 623-2341